# MCP361 Assignment6

Sanchit | 2021ME21063

## 1 Question 1

In this problem, we simulate the Multi-Armed Bandit with the following specifications:

- **Number of Arms**: The number of arms in the bandit.
- **Number of Iterations**: The number of independent simulations of the bandit problem.
- **Time Steps**: The number of time steps or rounds in each simulation.

The true reward for each arm follows a normal distribution with mean 0 and variance 1. The goal is to determine which arm provides the highest average reward over the simulation period.

### 1.1 Results Visualization

The results of the simulation are typically visualized using the plots

- The average reward per time step.
- The percentage of optimal actions taken over time.

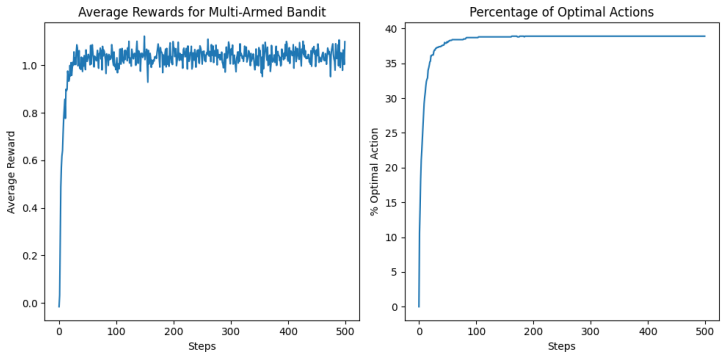


Figure 1: Simulation showing average reward and percentage of optimal actions

## 1.2 Pseudocode

**Algorithm 1** Multi-Armed Bandit Simulation

**Input:** Number of arms (`num_arms`), Number of iterations (`num_iterations`), Number of time steps (`time_steps`)
**Initialize:**
  True values of each arm `true_values` as a matrix of normal random variables
  Optimal arms `optimal_arms` based on true values
  Arm counts `arm_counts` initialized to 1 for each arm
  Estimated values `estimated_values` initialized to 0
**For each time step from 1 to `time_steps`:**
  Initialize `step_rewards` to store rewards for the current step
  Initialize `optimal_actions` to count optimal actions
  **For each iteration from 1 to `num_iterations`:**
    Choose the arm with the highest estimated value
    If chosen arm is optimal:
      Increment `optimal_actions`
    Generate reward for the chosen arm
    Update `arm_counts` and `estimated_values`
    Append reward to `step_rewards`
  **End For**
  **Calculate:**
    Average reward for the step
    Percentage of optimal actions
**End For**
**Output:** Average rewards and percentage of optimal actions over time

## 1.3 Explanation

This simulation investigates the effectiveness of a greedy strategy in solving the Multi-Armed Bandit problem. By executing multiple independent simulations, the algorithm's performance in identifying and exploiting the optimal arm is assessed. Each simulation involves iteratively selecting the arm with the highest estimated reward, updating the reward estimates, and tracking the frequency of optimal arm selections.

The pseudocode details the process of initializing the bandit arms, running the simulations, and computing crucial metrics, such as the average reward per time step and the percentage of times the optimal arm is chosen. The resulting plots provide insights into how well the greedy algorithm performs in discovering and maximizing rewards over time.

# 2 Question 2

For this problem, the modifications are done to the above code to run it for different values of $\epsilon$. The values of $\epsilon$ chosen are [0, 0.01, 0.05, 0.1, 0.25]. The results have been plotted below.

NOTE: $\epsilon = 0$ is essentially the greedy stratergy.

## 2.1 Results Visualization

The simulation results for different $\epsilon$ values is visually represented using the plots

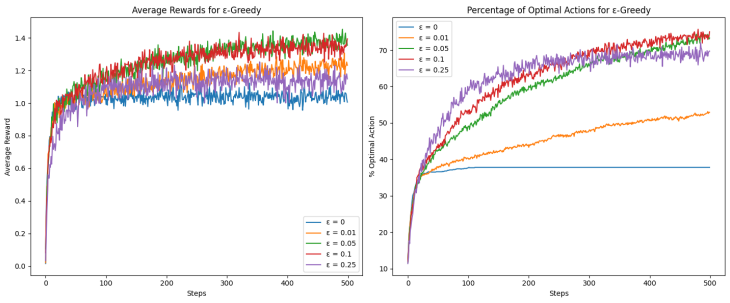


Figure 2: Simulation showing average reward and percentage of optimal actions for $\epsilon$ values.

## 2.2 Reason for Choosing $N = 1000$ Simulations and $T = 500$ Time Steps

The values $N = 1000$ simulations and $T = 500$ time steps were specified to ensure a thorough analysis while maintaining computational efficiency. Here's why these parameters are effective:

- $N = 1000$ **Simulations**: Conducting 1000 simulations provides a robust sample size that helps average out random fluctuations and ensures statistically significant results. This number of simulations offers a reliable assessment of the algorithm's performance across various bandit instances.
- $T = 500$ **Time Steps**: Utilizing 500 time steps per simulation allows sufficient time for the algorithm to effectively explore and exploit the arms. This duration is adequate to observe the learning and convergence behavior of the greedy algorithm, while still being computationally manageable.

These parameters offer a good balance between detailed analysis and practical computational constraints.

# 3 Comparative Insights

## 3.1 Variation of Average Rewards with Number of Steps

- The greedy algorithm, which always selects the arm with the highest estimated reward, generally yields lower average rewards in the long term compared to strategies that incorporate exploration (i.e., $\epsilon > 0$).
- A higher $\epsilon$ value, such as 0.25, leads to a lower average reward compared to a moderate $\epsilon$ of 0.1. This is because a higher $\epsilon$ results in more frequent exploration, which reduces the time spent exploiting the arm with the highest estimated reward, thereby decreasing the average reward.

## 3.2 Percentage of Times Optimal Actions Were Taken

- Over an extended period, the percentage of times the optimal action is chosen approaches $(1 - \epsilon) \times 100$. Therefore, a smaller $\epsilon$ results in a higher percentage of optimal actions in the limit.
- Achieving this high percentage, however, takes longer with smaller $\epsilon$ values because the algorithm explores less frequently. For example, with $\epsilon = 0.25$, the percentage of optimal actions increases quickly but plateaus at a lower value (around 75%). In contrast, with $\epsilon = 0.01$, the percentage rises more slowly but eventually reaches a higher value (close to 99%), eventually exceeding the performance of higher $\epsilon$ values over time.